〈経営研究　第16巻　第2号　平成14年12月〉

英書19冊の第2次および第3次語単位エントロピー計算

横　井　右　門

キーワード

エントロピー（平均情報量）entropy
アルファベット　alphabet
文字列エントロピー　string entropy
語単位エントロピー　word entropy
語彙　vocabulary
擬似コード　pseudo coding

はじめに

「エントロピー」という言葉は，まず熱力学で使われ，後にシャノンが始めたといわれる情報理論でシャノンが到達した結論であったが，その定義式が熱力学と同一であるためシャノンが「エントロピー」という言葉を当てはめたものである。これについてきわめて興味深いエピソードがある。

「ところで，シャノンは彼の式が示すこの概念に『エントロピー』という言葉を当てはめるのには気が進まなかったと伝えられる。しかし，フォン・ノイマンが，「誰もエントロピーなんかわかっちゃいないんだから，議論で

も君が先手なのははっきりしているよ」といって，いいからエントロピーという言葉を使えとすすめたらしい。」(1)

フォン・ノイマンのこのすすめは現在から考えると，少々余計なことをやってくれたものだと残念に思う。もしシャノンが「エントロピー entropy」ではなく，「平均情報量 average information」とでもしてくれれば現在にいたるまで続いている悲喜劇が発生しなかったはずである。「理科系」，「文科系」という分類は日本にしかないと聞くが，日本の大学教育において，ほとんどの理科系学部は熱力学を勉強し，熱力学には「熱力学の第二法則」なるものがあり，そこで熱力学のエントロピーを知識として身に付ける。一方，情報系の学部では暗号理論，符号理論等の前提として情報理論を学び，そこで情報理論のエントロピーを学ぶ。その情報系学部が理科系の場合，熱力学のエントロピーと情報理論のエントロピーは数式および名称が同じであっても，違うものだという認識に達するが，そうでない学部の卒業生，特に文科系に著しいが，その誤解が堂々とまかり通っているのが実情である。またエントロピーが計算可能な尺度であるという認識も極めて希薄である。また理科系卒業生についても情報系・通信工学系を除けば，情報理論のエントロピーは熱力学のそれと同じものだと錯覚している例が散見される。

1.「エントロピー」の使用例

最近まで蒐集した「エントロピー」という言葉が使用されている例を紹介する。

①「日本は年増加が 3 億トンの肥満国

日本の経済や社会の展望をエントロピー法則とのかかわりを通じて考えてみたい。最初に言えるのは，エントロピーは増大する一方の物理量だという点である。粗い言い方だが，これを社会・経済的な文脈に置き換えると，この世の中では廃熱や廃物が増える一方である，ということだ。」(2)

② 「エントロピーは人間たちにこの世の有限について教えたといわれるが，

ほとんどの人間にはその実感も乏しく人々は未だに地球や宇宙の広大さの迷妄から覚めることはない。」(3)

③「エントロピーの増大こそがエントロピーの減少なのだといわれても私にはまったくの珍紛漢である。」(4)

④「そして同一均質社会ほど，嫉妬エネルギーが増大するのだ。いわばエントロピー増大だ。」(5)

⑤「自然法則にはかなわないわけで地球を汚染する方向は簡単であるが，これを凝縮させるのは難しいのである。

コーヒーに砂糖を入れると溶解して拡がる。これはきわめて自然の流れであるが，逆に溶けた砂糖をコーヒーから抽出して戻すには大変なカネとエネルギーが必要だ。またコメ粒を握ってばら撒くのは一瞬で済むが，これをもとのように集めるのは大変な苦労が必要だ。かくのごとく，地球は分散化し，役に立たない資源が増えていき，ついには死滅していく。これがエントロピーに基づく地球の運命である。」(6)

⑥「いまや前人未踏の高みにのぼった始皇帝の得意や思うべし。なにごとにつけてもスケールの大きい始皇帝のことだから，その奢侈もまたはなはだしいかぎりだった。ただ始皇帝の場合は，エントロピーを暴発させ，長夜の宴でただらに富を蕩尽した殷の紂王とは異なり，その奢侈にも一種のダイナミズムが認められる。(7)

以上の例では熱力学のエントロピーを意識していると想像できるが，中には「エントロピー」という言葉を使う必要がないのではないかと思われるものもある。「エントロピー」を「エネルギー」に置き換えても文脈が崩れないのではないかと思うものもある。われわれの日常会話でも，よく耳にする事象であるが，「エネルギーの爆発」でも意味が通じるのに，「エントロピーの爆発」という発言をすることがある。このことをたしなめているのではないかと愚考するが，その来し方が熱力学にも情報理論にも関係がなかったと思われる山本夏彦が「エントロピー」がカタカナ語であることから，つぎの

ように述べているのは救いである。

「ユニオン・ショップとクローズド・ショップの違いも事典ではわからない。両方読んでもどこが違うか定かでない。オーディナリ・ピープルとコモン・ピープルの違いも分からない。共に，並，普通，尋常以下無数の字句が並んでいるが，その大半は同じだから酷似した言葉だと思うと，イギリス人はオーディナリと言われれば，喜ぶが，コモンと言われるとあなどられたと思う。コモンは一段下なのだそうだ。

かくて字引はそれを知らない人はもとより，すこし知るものにも役に立たない。そのことを字引の編者は知らなすぎる。エントロピーの如きはどの事典を見ても分からない。書いている当人が分かってない。分からないことを分かったふりして書くのはさぞ辛かろうと思うのに辛くない。」(8)

このまま放置するならば，40年も前にピアースが次のように解説したことが無駄だったことになる。

「この安直だが誤りを招きやすい観念は専門家の間にさえたくさんの混乱をひき起こした。本当はコミュニケーション理論は電気通信の分野のある種の問題を解くために生まれたものである。そのエントロピーは，統計力学のエントロピーとの数学的類似のためにエントロピーと名づけられたのである。このエントロピーが主に関係する問題は，統計力学が取り組む問題とはまったく異なるものである。」(9)

2. 本論の目的

エントロピーの計算例は非常に少ない。それはシャノンの最初の主たる発想は通信に関するものであり，今まで情報通信分野での研究が専らであったからである。エントロピーに関するシャノンの最初の論文名からも明らかである。(10) 通信機メーカーにしても，通信業者にしても通信情報関係の学部あ

るいは研究所には通信文書データも含む膨大なデータが存在する。

しかし，書籍等の磁気媒体等に変換された文書データは入手するのが非常に難しい。ワードプロセッサを日本で最初に開発したとされるメーカーが日本語データが少なく，当時それを磁気テープとして持っていた西日本の某国立大学に譲ってくれと申し入れ，結局断られたのは有名な話である。

計算例が少ないのは，データをまず作成しなければならないからである。スキャナーによるテキスト・データ作成がすぐ思いつかれるが，それがいかに困難なことか，それだけで一冊の論文が存在するほどである。[11] だからこそ，1955年にG.A.Barnardが Statistical Calculation of Word Entropies for Four European Languagesのデータが南によって引用され[12]，さらに藤田，堀によって[13][14]間接的に引用され，ヤグロムがより巧妙な形で引用している[15]原因であろう。近年では前川の第5次エントロピーまでの貴重な計算がある。[22] しかし，次数に対するエントロピーの減少速度が大きいところから，データ量が少ないことが類推できる。Barnardおよび前川の計算を孤証に終わらせたくなく，文字列エントロピーについて第15次エントロピーまでを19冊の英書について計算した。[16]

語単位エントロピーは前川の第1次エントロピー「2.23」というやはり貴重な計算があるだけである。これも同じ理由から同一テキストデータにより，19個の語単位第1次エントロピーを計算した。どれも2.23に近い値である。[17] しかし，この語単位エントロピーは第1次エントロピーにすぎないので，第2次，第3次エントロピーまで計算するのが本論の目的である。いままで計算に使用したテキストの平均単語長は3.9から4.7バイトである。[17] 第15次エントロピーは文字列長15字のテキストデータ・レコード・ファイルから計算される。平均単語長を仮に4.5バイトだとすると語単位第2次エントロピーの計算対象データの1レコードの長さはスペース1バイトを加え4.5+1+4.5で，約10バイト，語単位第3次エントロピーの計算対象データはスペース2個を加え，約14.5バイトと予想できる。第1次，第2次および第3次語単位エントロピーと文字列第1次エントロピーから文字列第15次

エントロピーまでとを比較するのが妥当である。

ハワード・ラインゴールドによれば，シャノンが情報理論の基礎を作り出した経緯について次のように述べている。[18] 少し長くなるが以下に，そのまま引用する。

情報理論の基礎は，一九四八年の二つの論文によって作り出された。そしてその中心となるのはボルツマンのエントロピーと，システムの秩序を結びつけた式との明確な関連をもつ基本的な式であった。しかしその式のもとになる基本的な考え方は単純なもので，
シャノンはコーディングと通信の定量的次元を理解する方法としてあるゲームを提案した。

そのゲームは「二〇の扉」のちょっとした変形である。英語のアルファベットの場合には五つの扉となる。ゲームをするうちの一人は，アルファベットの一文字を思い浮かべる。もう一方の人が質問してその文字を当てるのだが，質問は「その文字はアルファベット順でLよりも前にありますか」というようなものに限られる。これは厳密にイエス――ノーで答えられるゲームであり，一回毎にイエスかノーのどちらか一つで答える。

シャノンは英文に使われる三〇文字のうちの一つを決めるのに，最高で五つの質問が必要であるといっている。このイエスとノーの答えの組み合わせを，0と1との組み合わせやオンとオフの信号の組や，その他の二進法記号で表せば，アルファベット文字を表わす符号を得る――実際にそれはテレタイプライターの情報通信に用いる符号のもとになった。

このゲームは木構造として図示することができる。各文字はもともとは，幹から枝分かれした枝からさらに分かれた小枝につくたった一枚の葉である。あるいは，二股の小道のある庭園とみなすこともできよう。その庭園ではおのおのの小道が，二股の分岐点でどちらに行くかという決定の連続であり，どの終点の位置もその道筋をたどる決定の組み合わせによってコード化

することができる。それはまたコンピュータのメモリにアドレスを示したり，その場所に格納される命令をエンコードするよい方法にもなる。このゲーム―木―コードの基礎となる要素つまり二元的な決定は，シャノンの情報の基本的な単位「ビット」となった。コンピュータに夢中な人たちがビットについて語る時はいつでも，二股の小道がある庭園における判断の一つをさしているのである。

ビットが，小道の分岐や二〇の扉の数字や容器の中の分子のエネルギーの状態のどれかを示す時には，一つ一つが状況のあいまいさを減少させるものであることに注意してほしい。

以上であるが，「英文に使われる三〇文字のうちの一つを決めるのに，最高で五つの質問が必要である」について詳述する。前川　守「1000万人のコンピュータ科学　文章を科学する」で採用されている擬似コードを使用する。(19)

```
if（文字が m より後）
{
    if（文字が s より後）
    {
        if（文字が v より後）
        {
            if（文字が x より後）
            {
                if（文字が y より後）
                {
                    文字は z
                }
                else
                {
                    文字は y
                }
            }
            else
            {
                if（文字が w より後）
                {
                    文字は x
                }
                else
                {
                    文字は w
                }
            }
        }
        else
        {
            if（文字が t より後）
            {
                if（文字が u より後）
                {
                    文字は v
                }
                else
                {
                    文字は u
                }
            }
            else
            {
                文字は t
            }
        }
    }
    else
    {
        if（文字が p より後）
        {
            if（文字が r より後）
            {
```

```
                文字は s
            }
            else
            {
                if（文字が q より後）
                {
                    文字は r
                }
                else
                {
                    文字は q
                }
            }
        }
        else
        {
            if（文字が o より後）
            {
                文字は p
            }
            else
            {
                if（文字が n より後）
                {
                    文字は o
                }
                else
                {
                    文字は n
                }
            }
        }
    }
    else
    {
        if（文字が f より後）
        {
            if（文字が j より後）
            {
                if（文字が l より後）
                {
                    文字は m
                }
                else
                {
                    if（文字が k より後）
                    {
                        文字は l
                    }
                    else
                    {
                        文字は k
                    }
                }
            }
            else
            {
                if（文字が h より後）
                {
                    if（文字が i より後）
                    {
                        文字は j
                    }
                    else
                    {
                        文字は i
                    }
                }
                else
                {
                    if（文字が g より後）
                    {
                        文字は h
                    }
                    else
                    {
                        文字は g
                    }
                }
            }
        }
    }
    else
    {
        if（文字が c より後）
        {
            if（文字が e より後）
            {
                文字は f
            }
            else
            {
                if（文字が d より後）
                {
                    文字は e
                }
                else
                {
                    文字は d
                }
            }
        }
        else
        {
            if（文字が a より後）
            {
                if（文字が b より後）
                {
                    文字は c
                }
                else
```

```
    {
        文字はb
    }
}
if（文字が' ' より後）
{
    文字はa
}
```

```
                else
                {
                    文字は' '
                }
            }
        }
    }
}
```

以上の擬似c言語コーディングによるアルゴリズムは，少なくとも五つの質問によって，ある英字が何であるかを特定できる。5文字が4回の質問で特定でき，残りの22文字が5回の質問で特定できる。平均質問回数は4.81回である。$2^{4.81}$ は28.05である。$2^{4.7549}$ は27.00であるから，論理的な美しさにやや欠けるが，5回の質問でアルファベットを特定できる可能性を示している。4.7549は27字のアルファベットのchaos状態のエントロピー $\log_2 27$ に等しい。

3. 語単位第2次第3次エントロピー計算のための符号アルファベット

情報理論では符号アルファベットという言葉を使う。符号化により情報源資源に割り当てられた系列のそれぞれを符号語codewordと呼び，符号語すべての集合をcodeと呼ぶ。符号語に用いられる記号の集合を符号アルファベットと呼ぶ[20]。1ビットのとき符号アルファベットは{0, 1}である。前項の場合，符号アルファベットは{ , a, b, c, . . . , x, y, z}である。

文字列2次エントロピーのための2連字の符号アルファベットは{　, a, b, c, . . ., a ,aa, ab, . . . , zx, zy, zz}の 27^2 すなわち729文字である。3次エントロピーのための3連字は 27^3 すなわち19683文字からなる符号アルファベットである。10連字のときは，27^{10} 文字からなる符号アルファベットである。15連字のときは 27^{15} 文字からなる符号アルファベットである。JIS第一水準および第二水準を使った日本語の第1次エントロピーを計算しようとすれば，そのとき約7000字の符号アルファベットを使うことになる。日本語の第2次エントロピーを計算するときは，4千9百万字の符号アルファベットを使う

ことになる。

さて英語の語単位エントロピーを計算するときはどのように考えるべきか。

Audrey Hepburn が出演した My Fair Lady というミュージカルの名作に有名な次のフレーズが登場する。

The rain in Spain stays mainly in the plain.

大文字を小文字にし，ピリオドを省けば次のようになる。

the rain in spain stays mainly in the plain

ここで e rai という 5 連字の文字列と spain という 5 連字の文字列を比較した場合，文字列 e rai のほうが spain よりも，chaos に近く order からは遠いはずである。英単語の平均単語長を 5 として計算した 5 次エントロピーよりも語単位第 1 次エントロピーのほうが小さく，より order に近いはずである。これについては，すでに実証した[17]。語単位第 2 次エントロピーについては，この例文で the rain，rain in，in spain，spain stays，等についてスペースを含むそれを符号アルファベットと考えれば第 2 次語単位エントロピーが計算できる。the rain in，rain in spain，in spain stays，spain stays mainly，stays mainly in，mainly in the，in the plain のそれぞれを符号アルファベットとして計算すれば第 3 次語単位エントロピーが計算できる。そして求めたエントロピーをこの場合の符号アルファベットの平均長で除算すれば単位バイトあたりビット数のエントロピーが求められる。

4. データ

シャノンはエントロピー計算およびシミュレーションのための頻度を Secret and Urgent by Fletcher Pratt, Blue Ribbon Books, 1939 から求めた[21]。最初に 15 次エントロピーを計算した時，それに倣い，次の 19 冊の英書をデータとした[16]。当時，入力済みデータが当該 19 冊だったからである。すべて手作業入力によって作成した。その後，ブロック化エントロピー計算，語単

位エントロピー計算も同じデータを使ってきた。その延長で本稿においても同じデータを使用する。
著者名，書名，発行所，発行年を列挙する。

Tom Clancy

The Hunt for Red October: Harper Collins, 1993

Op-Center: Berkley Novels, 1995

Frederick Forsyth

The Day of Jackal: Bantam Books, 1995

The Dogs of War: Bantam Books, 1995

The Odessa File: Bantam Books, 1993

The Shepherd: Corgi Books, 1990

Akira Kohchi

Why I survived A-Bombs: Institute for Historical Reviews, 1989

Anne McCaffrey

The Crystal Singer: Corgi Books, 1991

Crystal Line: Del Rey Books, 1992

Robert B. Parker

A Catskill Eagle: Dell, 1985

Pastime: Berkley Novel, 1992

God Save the Child: Penguin Books, 1997

The Godwulf Manuscript: Dell, 1987

A. J. Quinnell

Man on Fire: Orion, 1994

In the Name of the Father: Signet Novel, 1987

The Blue Ring: Orion, 1994

Message from Hell: Orion, 1996

Jostein Gaarder

Sophie's World: Berkley Books, 1996

H. G. Wells

A Short History of the World: Collins, 1953

5. エントロピー計算システム

各書籍についてアルファベット以外の文字をすべてワードセパレータと考え，1単語1レコードからなるファイルを作成する。これを昇順でsortし，同一単語を累計し頻度を計算し，そして第1次エントロピーを計算する[17]。

さらに最初のファイルから2単語1レコードのファイルを作成し，昇順でsortし第1次語単位エントロピーと同様の計算をして第2次エントロピーを計算する。

最後に3単語1レコードのファイルを作成し，同様の処理で第3次エントロピーを計算する。システム・フローチャートは次のとおりである。「処理シンボル」中の名称はプログラム名である。(付録参照)

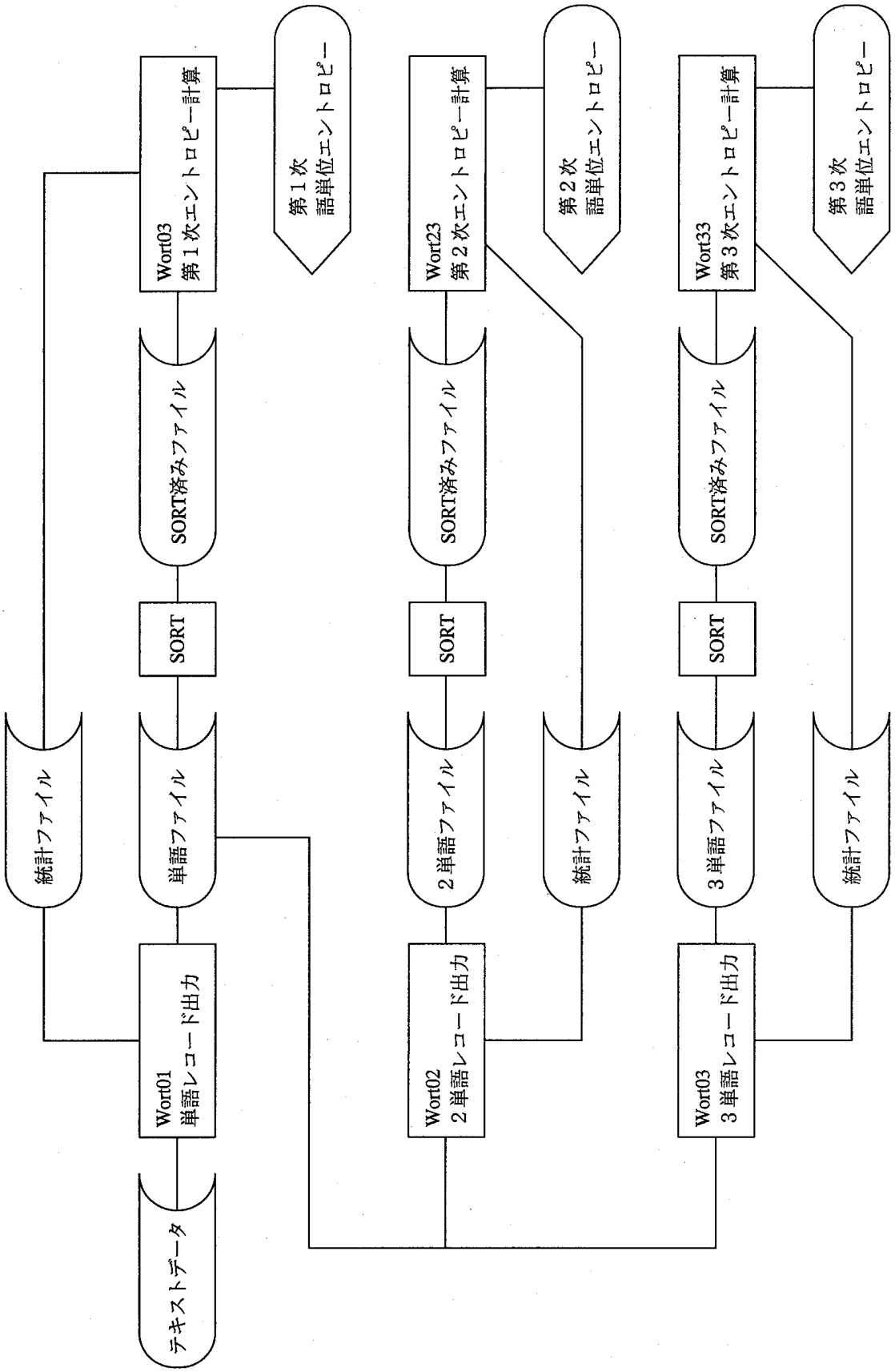

図１　語単位エントロピー計算システム

6. 計算結果

書名の下に第1次，第2次および第3次エントロピーHならびに対応する平均レコード長Lを記載した。エントロピーの略語としてHを用いる慣習はentropyのeに対応するギリシャ文字の大文字がHであることに由来する。その後に比較参照のため，既に発表してある[16]文字列の第15次までのエントロピーを列挙する。実際には有効数字7ケタであるが，ここでは有効数字3ケタとしておく。ただし書名は適当に縮めておいた。

The Hunt for Red October

	H	L
1	2.241969	4.396882
2	1.563801	9.739798
3	1.123380	15.109682

OP-Center

	H	L
1	2.309524	4.177216
2	1.570991	9.354487
3	1.116304	14.531710

The Day of Jackal

	H	L
1	2.231361	4.321260
2	1.541691	9.642555
3	1.112744	14.963862

The Dogs of War

	H	L
1	2.238117	4.316646
2	1.556018	9.633324
3	1.120716	14.950000

The Odessa File

	H	L
1	2.246768	4.242156
2	1.546479	9.484344
3	1.109337	14.726537

The Shepherd

	H	L
1	2.138244	4.191733
2	1.343135	9.383936
3	0.917231	14.575387

Why I survived A-Bomb

	H	L
1	2.191594	4.604874
2	1.439154	10.209863
3	0.996271	15.814808

The Crystal Singer

	H	L
1	2.178311	4.512841
2	1.493753	9.617797
3	1.060340	15.538625

Crystal Line

	H	L
1	2.241778	4.308889
2	1.528193	9.617797
3	1.083008	14.926701

A Catskill Eagle

	H	L
1	2.338234	3.922401
2	1.582873	8.884483
3	1.140106	13.767264

英書 19 冊の第 2 次および第 3 次語単位エントロピー計算

Pastime

	H	L
1	2.326463	3.901163
2	1.564540	8.802342
3	1.117369	13.703554

God Save the Child

	H	L
1	2.346035	3.918341
2	1.570637	8.836786
3	1.118771	13.755282

The Godwulf Manuscript

	H	L
1	2.339685	3.916888
2	1.575399	8.833790
3	1.122460	13.750742

Man on Fire

	H	L
1	2.243182	4.233800
2	1.544474	9.457606
3	1.103861	14.701395

In the Name of the Father

	H	L
1	2.252796	4.242949
2	1.552791	9.485902
3	1.114112	14.728868

The Blue Ring

	H	L
1	2.231504	4.193604
2	1.549920	9.387213
3	1.119356	14.580771

Message from Hell

	H	L
1	2.248351	4.119280
2	1.544315	9.238563
3	1.110879	14.357895

Sophie's World

	H	L
1	2.226911	4.286571
2	1.561186	9.573156
3	1.132969	14.859769

A Short History of the World

	H	L
1	2.024659	4.766497
2	1.414788	10.533050
3	1.012218	13.299616

第 1 次～第 15 次エントロピーの表

書　名	H01	H02	H03	H04	H05	H06	H07	H08	H09	H10	H11	H12	H13	H14	H15
Red October	4.08	3.74	3.41	3.10	2.82	2.59	2.38	2.20	2.03	1.87	1.73	1.61	1.49	1.40	1.31
OP Center	4.06	3.71	3.38	3.07	2.80	2.56	2.33	2.14	1.96	1.81	1.67	1.54	1.43	1.34	1.25
Jackal	4.06	3.70	3.36	3.06	2.79	2.56	2.35	2.13	1.99	1.84	1.70	1.58	1.47	1.37	1.29
Dogs of War	4.07	3.70	3.38	3.07	2.80	2.57	2.36	2.17	2.00	1.85	1.71	1.59	1.48	1.38	1.29
Odessa	4.06	3.71	3.37	3.06	2.79	2.55	2.34	2.14	1.97	1.82	1.68	1.56	1.45	1.35	1.26
Shepherd	4.08	3.71	3.35	2.98	2.65	2.36	2.11	1.89	1.71	1.56	1.42	1.31	1.21	1.13	1.05
A-Bomb	4.11	3.75	3.44	3.13	2.84	2.58	2.35	2.13	1.95	1.78	1.64	1.51	1.40	1.31	1.22
Singer	4.09	3.73	3.39	3.07	2.80	2.56	2.35	2.16	1.98	1.83	1.69	1.56	1.45	1.36	1.27
Crystal	4.07	3.72	3.38	3.06	2.78	2.55	2.33	2.14	1.96	1.80	1.66	1.54	1.43	1.33	1.25

Catskill	4.03	3.67	3.32	3.00	2.72	2.48	2.26	2.07	1.90	1.75	1.62	1.50	1.39	1.30	1.21
Pastime	4.04	3.68	3.32	3.00	2.71	2.47	2.24	2.05	1.87	1.72	1.59	1.47	1.36	1.27	1.16
Child	4.05	3.70	3.35	3.03	2.75	2.49	2.27	2.07	1.89	1.74	1.60	1.48	1.37	1.28	1.20
Godwulf	4.05	3.70	3.35	3.03	2.75	2.50	2.27	2.08	1.90	1.74	1.60	1.48	1.38	1.28	1.20
Man on Fire	4.06	3.70	3.37	3.05	2.78	2.54	2.33	2.14	1.97	1.81	1.67	1.55	1.44	1.34	1.26
Father	4.07	3.71	3.38	3.07	2.79	2.55	2.34	2.15	1.98	1.82	1.69	1.56	1.45	1.36	1.27
Blue Ring	4.06	3.70	3.36	3.04	2.77	2.53	2.32	2.20	1.96	1.81	1.67	1.55	1.44	1.35	1.26
Hell	4.06	3.70	3.35	3.03	2.75	2.50	2.29	2.10	1.93	1.78	1.64	1.52	1.42	1.32	1.24
Sophie	4.05	3.70	3.37	3.05	2.78	2.55	2.35	2.16	2.00	1.85	1.71	1.59	1.49	1.39	1.30
History	4.08	3.71	3.38	3.07	2.78	2.54	2.33	2.14	1.98	1.83	1.69	1.57	1.46	1.37	1.28

7. 結論

折れ線グラフを描いて見れば，一目瞭然であるが，文字列についての第1次エントロピーから第15次エントロピーまでのグラフと比較したとき，単語についての第1次エントロピーから第3次エントロピーははるかに下方に存在する。すなわち文字列と単語列とでは，単語列の依存関係がはるかに強力であることが数値的に判断できる。前川守教授が「語単位で，すなわち意味のある文字列で考えた方が，1文字あたりのエントロピーは大きく減少する」[(22)] としている見解が，第2次，第3次語単位エントロピーではさらに大きくあてはまることになる。本稿で取り上げた19冊のいずれもこの見解を裏付けているが，紙数上6冊のグラフを例示する。

特にThe Shepherdのエントロピーの収束速度が早いことに注意されたい。The Shepherdはわずか64kバイトであるのに対し，他のファイルは300kバイトから950kバイトの大きさである。

英書19冊の第2次および第3次語単位エントロピー計算

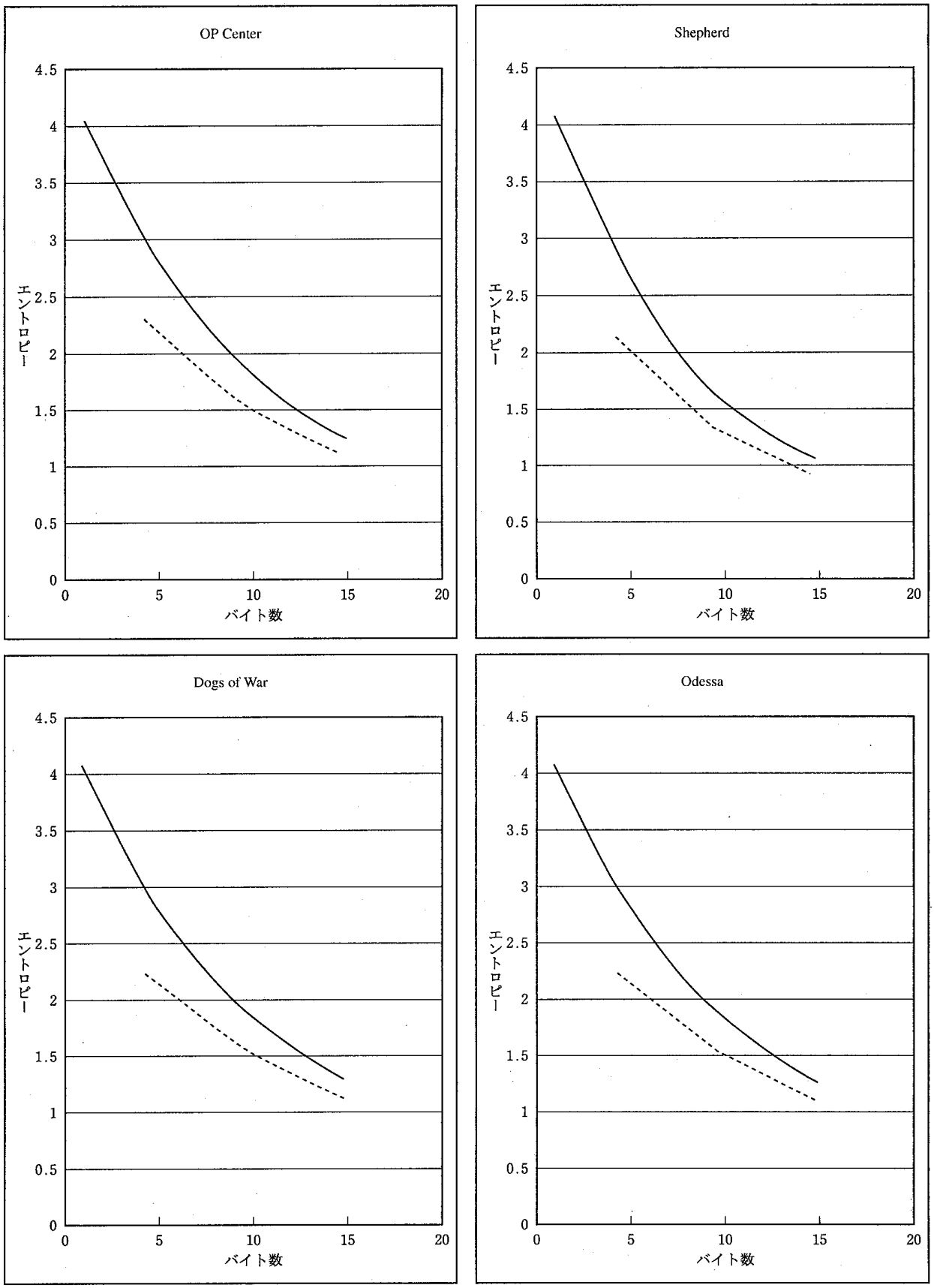

図2　文字列エントロピーと語単位エントロピーの収束状況

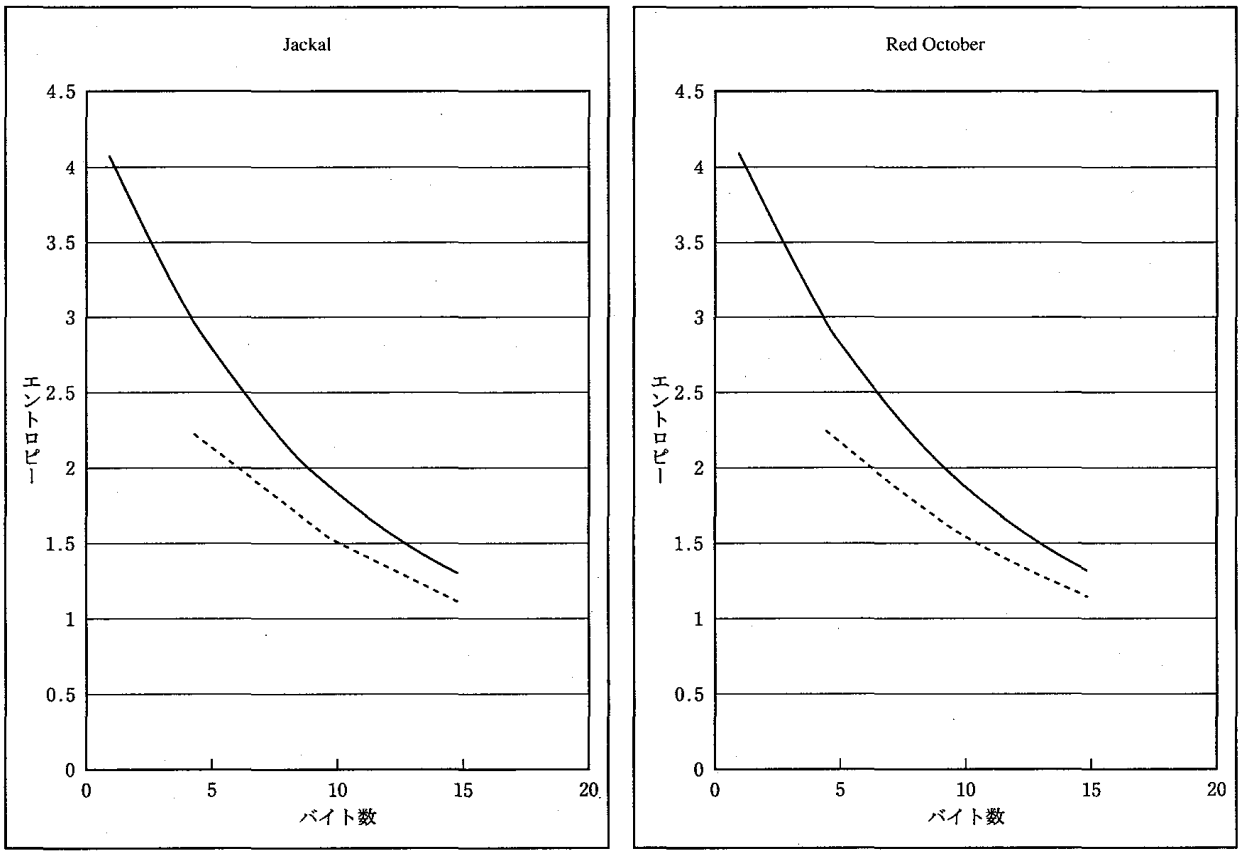

図２　文字列エントロピーと語単位エントロピーの収束状況

参考文献

(1) ハワード・ラインゴールド（栗田昭平・青木真美訳）：思考のための道具，パーソナルメディア（1995），p164
(2) 室田　武：もっとエントロピーに目を，日本経済新聞（2000.8.23）
(3) 石原慎太郎：国家なる幻想，文藝春秋 97 年 2 月号，㈱文藝春秋
(4) 西部　邁：改革のエントロピー，正論 97 年 2 月号，産経新聞社
(5) 早坂　暁：花へんろ通信，週刊新潮 97 年 1 月 2 日号，新潮社
(6) 牧野　昇：日本社会を襲う四つの危機，Voice 1997 年 12 月号，PHP 研究所
(7) 井波律子：酒池肉林，講談社，（1993）
(8) 山本夏彦：豆朝日新聞，文春文庫（1995）
(9) J.R.Pierce: Symbols, Signals and Noise, Harper & Row（1961），p23
(10) C.E.Shannon: The Mathematical Theory of Communication, Bell System Technical Journal, July and October, 1948
(11) 岡田　毅：実践コンピュータ英語学，鶴見書店（1995）
(12) 南　敏：情報理論，産業図書（1995），p55
(13) 藤田広一：情報理論，昭晃堂（1996）,p24
(14) 堀　淳一：エントロピーとは何か，講談社（1994），p194
(15) ヤグロム（井関，西田訳）：情報理論入門，みすず書房（1958）
(16) 横井右門：英書 19 冊の 15 次エントロピーの計算，経営研究　第 13 巻第 2 号，愛知学泉大学経営研究所（1999）
(17) 横井右門：英書 19 冊の語単位エントロピー計算，経営研究　第 16 巻第 1 号，愛知学泉大学経営研究所（2002）
(18) ハワード・ラインゴールド（栗田昭平・青木真美訳）：思考のための道具，パーソナルメディア（1995），p160
(19) 前川　守：文章を科学する，岩波書店（1995）p48
(20) 今井秀樹：情報理論，昭晃堂（2000），p11
(21) C.E.Shannon: The Mathematical Theory of Communication, Bell System Technical Journal, July and October, 1948, p42
(22) 前川　守：文章を科学する，岩波書店（1995）p87

付録１

```
001:/* Wort01.c Word Record Output */
002:#include <stdio.h>
003:void main(void)
004:{
005:    int i = 0, ccnt = 0, wordcnt = 0;
006:    float ave;
007:    char chara, work[100], fname1[50], fname2[50];
008:    FILE * fp1, * fp2, * fp3;
009:    printf("File1:");
010:    scanf("%s", fname1);
011:    printf("File2:");
012:    scanf("%s", fname2);
013:    fp1 = fopen(fname1, "r");
014:    fp2 = fopen(fname2, "w");
015:    chara = getc(fp1);
016:    while(chara != EOF)
017:    {
018:        if(65 <= chara && chara <= 90)
019:        {
020:            chara = chara + 32;
021:        }
022:        if(97 <= chara && chara <= 122)
023:        {
024:            work[i] = chara;
025:            i = i + 1;
026:        }
027:        else
028:        {
029:            if(i == 0)
030:            {
031:            }
032:            else
033:            {
034:                ccnt = ccnt + i;
035:                wordcnt = wordcnt + 1;
036:                work[i] = '\0';
037:                i = 0;
038:                fputs(work, fp2);
039:                fputc('\n', fp2);
040:            }
041:        }
042:        chara = getc(fp1);
043:    }
044:    ccnt = ccnt + i;
045:    wordcnt = wordcnt + i;
046:    work[i] = '\0';
047:    fputs(work, fp2);
048:    fputc('\0', fp2);
049:    ave = (float)ccnt / (float)wordcnt;
050:    printf("%d words  %7d bytes  ave: %8.6f\n", wordcnt, ccnt, ave);
051:    fclose(fp1);
052:    fclose(fp2);
053:    fp3 = fopen("statis.txt", "w");
054:    fprintf(fp3, "%7d %7d\n", wordcnt, ccnt);
055:    fclose(fp3);
056:}
```

付録 2

```
001:/* Wort03.c  語単位エントロピー計算 */
002:#include<stdio.h>
003:#include <string.h>
004:#include <math.h>
005:void main(void)
006:{
007:   char work[100], trace[100], * killa,
008:        fname1[50];
009:   int n, in_cnt = 0, out_cnt = 0, frequency = 0, words, characters;
010:   double proba, average, entropy = 0, selfinfo, entropybyte;
011:   FILE * fp1, * fp3;
012:   printf("in file 1: ");
013:   scanf("%s", fname1);
014:   fp1 = fopen(fname1, "r");
015:   fp3 = fopen("statis.txt", "r");
016:   fscanf(fp3, "%7d %7d", &words, &characters);
017:   average = (float)characters / (float)words;
018:   printf("(Wort03) ");
019:   printf("%7d words, %d characters average = %8.6f\n",
020:           words, characters, average);
021:   killa = fgets(work, 100, fp1);
022:   strcpy(trace, work);
023:   while(killa != NULL)
024:   {
025:      in_cnt = in_cnt + 1;
026:      frequency = frequency + 1;
027:      n = strcmp(work, trace);
028:      if (n > 0)
029:      {
030:         proba = (float)frequency / (float)words;
031:         selfinfo = log(proba) / log(2.0);
032:         entropy = entropy + proba * selfinfo;
033:         out_cnt = out_cnt + 1;
034:         frequency = 0;
035:         strcpy(trace, work);
036:
037:      }
038:      killa = fgets(work, 100, fp1);
039:   }
040:   frequency = frequency + 1;
041:   proba = (float)frequency / (float)words;
042:   selfinfo = log(proba) / log(2.0);
043:   entropy = entropy + proba * selfinfo;
044:   entropybyte = entropy / average;
045:   printf("%8.6f %8.6f %8.6f\n", entropy, average, entropybyte);
046:   out_cnt = out_cnt + 1;
047:
048:   printf("in %d records, out %d records\n", in_cnt, out_cnt);
049:   fclose(fp1);
050:   fclose(fp3);
051:}
052:
```

付録３

```
001:/* Wort21.c ２語レコード出力 */
002:#include <stdio.h>
003:#include <string.h>
004:#include <math.h>
005:void main(void)
006:{
007:    char work[100], first[50], * killa, fname1[50], fname2[50],
008:         longstring[150], longestimage[150];
009:    int i, l = 0, longest = 0,
010:       recordcnt = 0, lengthcnt = 0, freq[150];
011:    double ave, devia, vari = 0, stdevia;
012:    FILE * fp1, * fp2, * fp3;
013:    for(i = 0; i < 150; i++)
014:    {
015:        freq[i] = 0;
016:    }
017:    printf(" in file:");
018:    scanf("%s", fname1);
019:    printf("out file:");
020:    scanf("%s", fname2);
021:    fp1 = fopen(fname1, "r");
022:    fp2 = fopen(fname2, "w");
023:    fp3 = fopen("statis.txt", "w");
024:    killa = fgets(work, 100, fp1);
025:    for(i = 0; work[i] != 0x0a; i++)
026:    {
027:        longstring[l] = work[i];
028:        l++;
029:    }
030:    longstring[l] = ' ';
031:    l++;
032:    killa = fgets(work, 100, fp1);
033:    for(i = 0; work[i] != 0x0a; i++)
034:    {
035:        longstring[l] = work[i];
036:        l++;
037:    }
038:    longstring[l] = '\0';
039:    fputs(longstring, fp2);
040:    fputc('\n', fp2);
041:    if(l > longest)
042:    {
043:        longest = l;
044:    }
045:    strcpy(first, work);
046:    strcpy(longestimage, longstring);
047:    recordcnt = recordcnt + 1;
048:    lengthcnt = lengthcnt + l;
049:    freq[l] = freq[l] + 1;
050:    l = 0;
051:    killa = fgets(work, 100, fp1);
052:    while(killa != NULL)
053:    {
054:        for(i = 0; first[i] != 0x0a; i++)
055:        {
056:            longstring[l] = first[i];
057:            l++;
058:        }
059:        longstring[l] = ' ';
060:        l++;
061:        for(i = 0; work[i] != 0x0a; i++)
062:        {
063:            longstring[l] = work[i];
064:            l++;
065:        }
066:        longstring[l] = '\0';
067:        fputs(longstring, fp2);
068:        fputc('\n', fp2);
069:        if(l > longest)
070:        {
071:            longest = l;
```

```
072:            strcpy(longestimage, longstring);
073:        }
074:        recordcnt = recordcnt + 1;
075:        lengthcnt = lengthcnt + l;
076:        freq[l] = freq[l] + 1;
077:        strcpy(first, work);
078:        l = 0;
079:        killa = fgets(work, 100, fp1);
080:    }
081:    for(i = 0; i <= longest; i++)
082:    {
083:        if(i % 5 == 0)
084:        {
085:            printf("\n");
086:        }
087:        printf("%2d -> %6d | ", i, freq[i]);
088:    }
089:    ave = (float)lengthcnt / (float)recordcnt;
090:    for(i = 0; i <= longest; i++)
091:    {
092:        devia = (float) i - ave;
093:        vari = vari + devia * devia * (float)freq[i];
094:    }
095:    vari = vari / (float)recordcnt;
096:    stdevia = sqrt(vari);
097:    printf("\n");
098:    printf("average = %8.6f, standard deviation = %8.6f\n",
099:                ave, stdevia);
100:    printf("lengthcnt = %7d, recordcnt = %7d\n",
101:                lengthcnt, recordcnt);
102:    printf("longest: %d bytes, longest word group: %s\n",
103:                longest, longestimage);
104:    fprintf(fp3, "%7d%7d\n", lengthcnt, recordcnt);
105:    fclose(fp1);
106:    fclose(fp2);
107:    fclose(fp3);
108:}
```

付録 4

```
001:/* 第２次語単位エントロピー計算 */
002:#include <stdio.h>
003:#include <string.h>
004:#include <math.h>
005:void main(void)
006:{
007:    char work[100], trace[100], * killa,
008:        fname1[50];
009:    int n, in_cnt = 0, out_cnt = 0, frequency = 0, lengthcnt, recordcnt;
010:    double proba, average, entropy = 0, selfinfo, byteentropy;
011:    FILE * fp1, * fp3;
012:    printf("in file: ");
013:    scanf("%s", fname1);
014:    fp1 = fopen(fname1, "r");
015:    fp3 = fopen("statis.txt", "r");
016:    fscanf(fp3, "%7d%7d", &lengthcnt, &recordcnt);
017:    average = (float)lengthcnt / (float)recordcnt;
018:    printf("sum of length:%7d, %7d records, average: %8.6f\n",
019:           lengthcnt, recordcnt, average);
020:    killa = fgets(work, 100, fp1);
021:    strcpy(trace, work);
022:    while(killa != NULL)
023:    {
024:        in_cnt = in_cnt + 1;
025:        frequency = frequency + 1;
026:        n = strcmp(work, trace);
027:        if(n > 0)
028:        {
029:            proba = (float)frequency / (float)recordcnt;
030:            selfinfo = log(proba) / log(2.0);
031:            entropy = entropy + proba * selfinfo;
032:            out_cnt = out_cnt + 1;
033:            frequency = 0;
034:            strcpy(trace, work);
035:        }
036:        killa = fgets(work, 100, fp1);
037:    }
038:    frequency = frequency + 1;
039:    proba = (float)frequency / (float)recordcnt;
040:    selfinfo = log(proba) / log(2.0);
041:    entropy = entropy + proba * selfinfo;
042:    byteentropy = entropy / average;
043:    printf("%8.6f %8.6f %8.6f\n", entropy, average, byteentropy);
044:    out_cnt = out_cnt + 1;
045:    printf("in %d records, out %d records\n", in_cnt, out_cnt);
046:    fclose(fp1);
047:    fclose(fp3);
048:}
049:
050:
```

付録 5

```
001:/* Wort31.c 3語レコード出力 */
002:#include <stdio.h>
003:#include <string.h>
004:#include <math.h>
005:void main(void)
006:{
007:    char work[100], first[50], second[50], * killa, fname1[50], fname2
[50],
008:         longstring[150], longestimage[150];
009:    int i, l = 0, longest = 0,
010:        recordcnt = 0, lengthcnt = 0, freq[150];
011:    double ave, devia, vari = 0, stdevia;
012:    FILE * fp1, * fp2, * fp3;
013:    for(i = 0; i < 150; i++)
014:    {
015:        freq[i] = 0;
016:    }
017:    printf(" in file:");
018:    scanf("%s", fname1);
019:    printf("out file:");
020:    scanf("%s", fname2);
021:    fp1 = fopen(fname1, "r");
022:    fp2 = fopen(fname2, "w");
023:    fp3 = fopen("statis.txt", "w");
024:    killa = fgets(work, 100, fp1);
025:    for(i = 0; work[i] != 0x0a; i++)
026:    {
027:        longstring[l] = work[i];
028:        l++;
029:    }
030:    longstring[l] = ' ';
031:    l++;
032:    killa = fgets(work, 100, fp1);
033:    for(i = 0; work[i] != 0x0a; i++)
034:    {
035:        longstring[l] = work[i];
036:        l++;
037:    }
038:    strcpy(first, work);
039:    longstring[l] = ' ';
040:    l++;
041:    killa = fgets(work, 100, fp1);
042:    for(i = 0; work[i] != 0x0a; i++)
043:    {
044:        longstring[l] = work[i];
045:        l++;
046:    }
047:    strcpy(second, work);
048:    longstring[l] = '\0';
049:    fputs(longstring, fp2);
050:    fputc('\n', fp2);
051:    if(l > longest)
052:    {
053:        longest = l;
054:    }
055:    strcpy(longestimage, longstring);
056:    recordcnt = recordcnt + 1;
057:    lengthcnt = lengthcnt + l;
058:    freq[l] = freq[l] + 1;
059:    l = 0;
060:    killa = fgets(work, 100, fp1);
061:    while(killa != NULL)
062:    {
063:        for(i = 0; first[i] != 0x0a; i++)
064:        {
065:            longstring[l] = first[i];
066:            l++;
067:        }
068:        longstring[l] = ' ';
069:        l++;
070:        for(i = 0; second[i] != 0x0a; i++)
```

```
071:        {
072:            longstring[l] = second[i];
073:            l++;
074:        }
075:        longstring[l] = ' ';
076:        l++;
077:        for(i = 0; work[i] != 0x0a; i++)
078:        {
079:            longstring[l] = work[i];
080:            l++;
081:        }
082:        longstring[l] = '\0';
083:        fputs(longstring, fp2);
084:        fputc('\n', fp2);
085:        if(l > longest)
086:        {
087:            longest = l;
088:            strcpy(longestimage, longstring);
089:        }
090:        recordcnt = recordcnt + 1;
091:        lengthcnt = lengthcnt + l;
092:        freq[l] = freq[l] + 1;
093:        strcpy(first, second);
094:        strcpy(second, work);
095:        l = 0;
096:        killa = fgets(work, 100, fp1);
097:    }
098:    for(i = 0; i <= longest; i++)
099:    {
100:        if(i % 5 == 0)
101:        {
102:            printf("\n");
103:        }
104:        printf("%2d -> %6d | ", i, freq[i]);
105:    }
106:    ave = (float)lengthcnt / (float)recordcnt;
107:    for(i = 0; i <= longest; i++)
108:    {
109:        devia = (float) i - ave;
110:        vari = vari + devia * devia * (float)freq[i];
111:    }
112:    vari = vari / (float)recordcnt;
113:    stdevia = sqrt(vari);
114:    printf("\n");
115:    printf("average = %8.6f, standard deviation = %8.6f\n",
116:                ave, stdevia);
117:    printf("lengthcnt = %7d, recordcnt = %7d\n",
118:                  lengthcnt, recordcnt);
119:    printf("longest: %d bytes, longest word group: %s\n",
120:                    longest, longestimage);
121:    fprintf(fp3, "%7d%7d\n", lengthcnt, recordcnt);
122:    fclose(fp1);
123:    fclose(fp2);
124:    fclose(fp3);
125:}
```

付録6

```
001:/* wort33.c 第3次語単位エントロピー計算 */
002:#include <stdio.h>
003:#include <string.h>
004:#include <math.h>
005:void main(void)
006:{
007:    char work[100], trace[100], * killa,
008:        fname1[50];
009:    int n, in_cnt = 0, out_cnt = 0, frequency = 0, lengthcnt, recordcnt;
010:    double proba, average, entropy = 0, selfinfo, byteentropy;
011:    FILE * fp1, * fp3;
012:    printf("in file: ");
013:    scanf("%s", fname1);
014:    fp1 = fopen(fname1, "r");
015:    fp3 = fopen("statis.txt", "r");
016:    fscanf(fp3, "%7d%7d", &lengthcnt, &recordcnt);
017:    average = (float)lengthcnt / (float)recordcnt;
018:    printf("sum of length:%7d, %7d records, average: %8.6f\n",
019:           lengthcnt, recordcnt, average);
020:    killa = fgets(work, 100, fp1);
021:    strcpy(trace, work);
022:    while(killa != NULL)
023:    {
024:        in_cnt = in_cnt + 1;
025:        frequency = frequency + 1;
026:        n = strcmp(work, trace);
027:        if(n > 0)
028:        {
029:            proba = (float)frequency / (float)recordcnt;
030:            selfinfo = log(proba) / log(2.0);
031:            entropy = entropy + proba * selfinfo;
032:            out_cnt = out_cnt + 1;
033:            frequency = 0;
034:            strcpy(trace, work);
035:        }
036:        killa = fgets(work, 100, fp1);
037:    }
038:    frequency = frequency + 1;
039:    proba = (float)frequency / (float)recordcnt;
040:    selfinfo = log(proba) / log(2.0);
041:    entropy = entropy + proba * selfinfo;
042:    byteentropy = entropy / average;
043:    printf("%8.6f %8.6f %8.6f\n", entropy, average, byteentropy);
044:    out_cnt = out_cnt + 1;
045:    printf("in %d records, out %d records\n", in_cnt, out_cnt);
046:    fclose(fp1);
047:    fclose(fp3);
048:}
```